Algorithmen

Inf-Einf-B · Vorlesung 3, erster Teil

Prof. Dr. Dominik Herrmann

Einführung in die Informatik (Inf-Einf-B)

Otto-Friedrich-Universität Bamberg

WiSe 2026/27

Pause · bis 16:15 Uhr

Aus dem letzten Jahrgang

**Ohne Vorerfahrung, viel KI:
1,4 von 5 Übungsaufgaben
ohne Hilfe zugetraut. Wenig KI: 2,9.**

Was davon Ursache ist, wissen wir nicht.
Versuchen Sie Cäsar erst selbst.

Wie lange dauert das?

Heute

Suchen	linear und binär
Laufzeit	O und Ω
Schleifen	`break` und `continue`
Eigene Datentypen	`struct`

Wie oft wird `strlen` aufgerufen?

```
string s = "Hallo";
for (int i = 0; i < strlen(s); i++)
{
    printf("%c", s[i]);
}
```

6-mal: Die Bedingung wird vor jedem Durchlauf geprüft, auch vor dem, der nicht mehr stattfindet.

Ein langer String


```
char s[4000001];
...
int zaehler = 0;
for (int i = 0; i < strlen(s); i++)
{
    if (s[i] == 'a')
    {
        zaehler++;
    }
}
```

```
$ time ./laenge 250000
250000

real    0m0.595s
```

Wie lange dauert eine Million?

Zeichen	Zeit
250 000	0,6 s
500 000	2,4 s
1 000 000	9,6 s
2 000 000	etwa 38 s

Die Länge einmal vorher


```
for (int i = 0, n = strlen(s); i < n; i++)
```

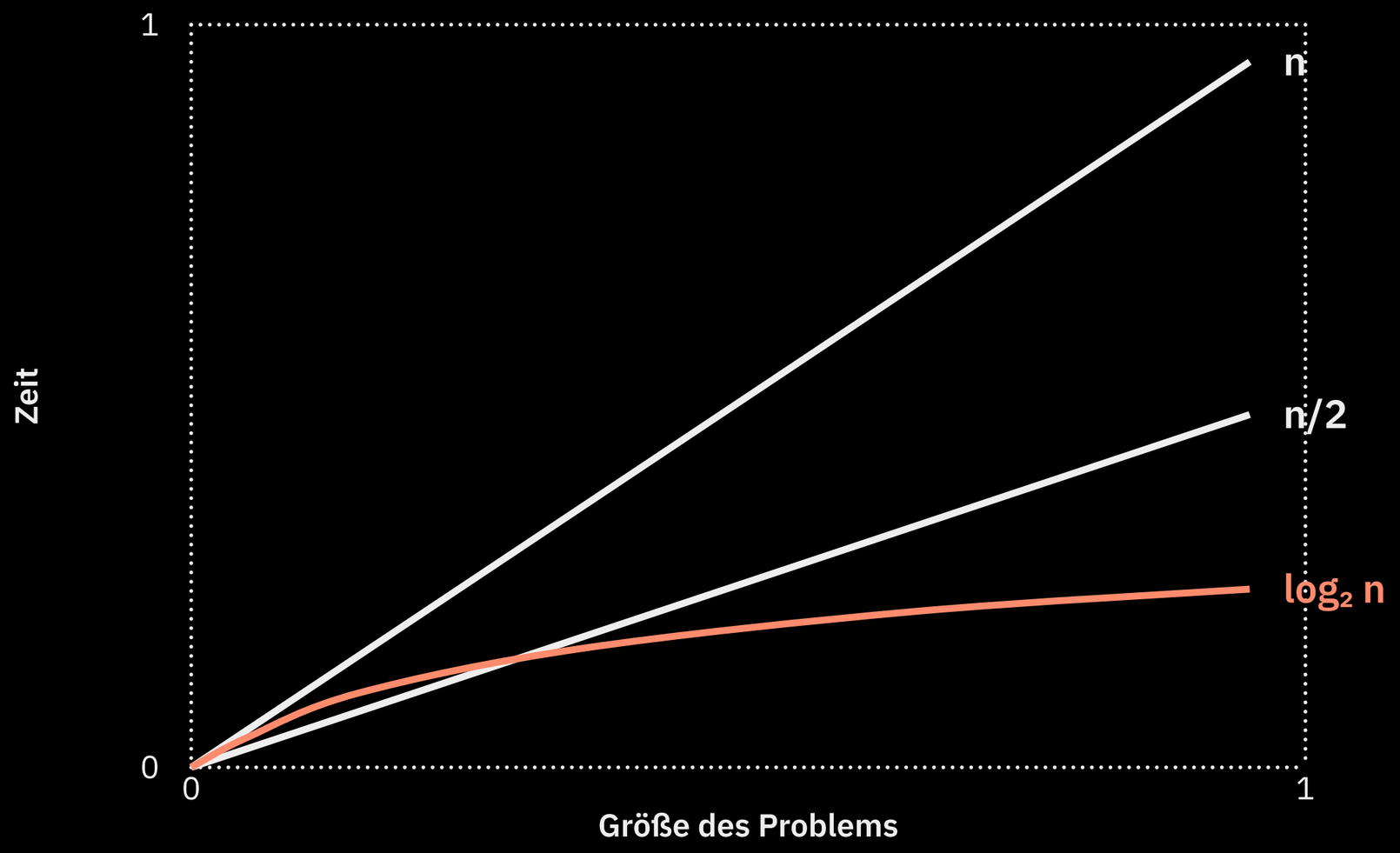
Erster Teil: läuft einmal und legt `i` und n an.

Bedingung: vor jedem Durchlauf, nur noch `i < n`.

```
$ time ./laenge1 2000000
2000000

real    0m0.003s
```

Logarithmisch zählen


1
0
0
1
Zeit
Größe des Problems
n
n/2
log₂ n

Wie viele sind wir?

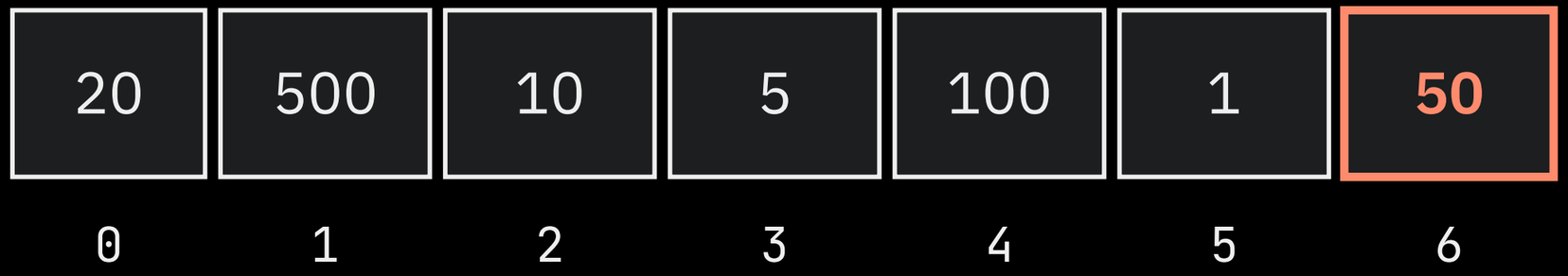
1. Stehen Sie auf und denken Sie an die Zahl 1.
2. Bilden Sie ein Paar mit jemandem, der steht. Addieren Sie beide Zahlen, merken Sie sich die Summe.
3. Eine Person des Paares setzt sich.
4. Stehen Sie noch? Zurück zu 2.

Jede Runde halbiert.

110 Personen: etwa **7 Runden** ($2^7 = 128$). Doppelt so viele: **eine** Runde mehr.

Lineare Suche

Ist die 50 dabei?



Eingabe: Kisten und 50 → Suche → true oder false

Lineare Suche als Pseudocode

```
Für jede Kiste von links nach rechts
    Wenn 50 in der Kiste ist
        Gib true zurück
Gib false zurück

    Für i von 0 bis n-1
        Wenn 50 in kisten[i] ist
            Gib true zurück
    Gib false zurück
```

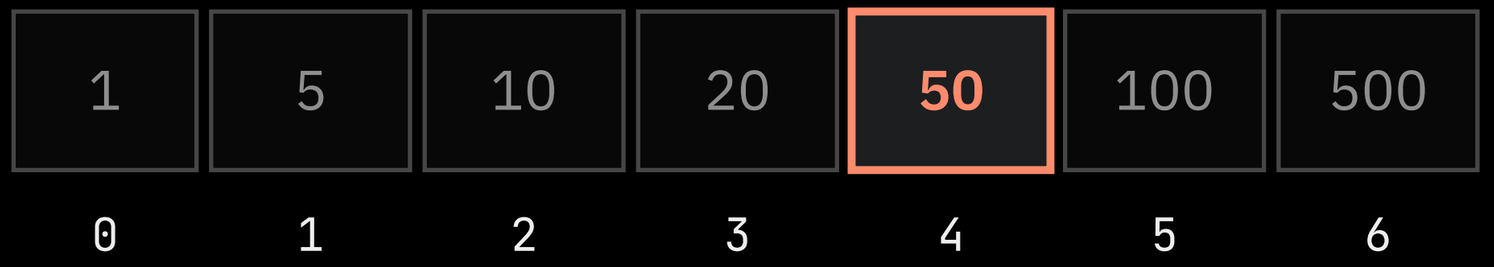
Was ist hier falsch?

```
Für jede Kiste von links nach rechts
    Wenn 50 in der Kiste ist
        Gib true zurück
    Sonst
        Gib false zurück
```

Gibt nach der ersten Kiste auf: `Gib false zurück` gehört hinter die Schleife.

Binäre Suche

gefunden, 3 Kisten



Binäre Suche als Pseudocode


```
Wenn 50 in der mittleren Kiste ist
    Gib true zurück
Sonst wenn 50 < mittlere Kiste
    Durchsuche die linke Hälfte
Sonst wenn 50 > mittlere Kiste
    Durchsuche die rechte Hälfte

Wenn keine Kisten übrig sind
    Gib false zurück
Wenn 50 in der mittleren Kiste ist
    Gib true zurück
Sonst wenn 50 < mittlere Kiste
    Durchsuche die linke Hälfte
Sonst wenn 50 > mittlere Kiste
    Durchsuche die rechte Hälfte
```

Wo ist die 50?

Noch einmal. Die Kisten sind wieder sortiert.

Sie waren es nicht.

Die binäre Suche prüft das nie, sie vertraut Ihnen.

Jetzt Sie: Binäre Suche von Hand

3 Minuten, auf Papier, Laptop zu. Wählen Sie die Stufe, die Sie schaffen.

Stufe 1 `3 8 15 23 42 57 91` Gesucht: 15. Welche Zahlen sehen Sie nacheinander an?

Stufe 2 `2 4 7 11 16 22 29 37 46 56 67 79 92 106 121` Gesucht: 40.
Welche Zahlen, und woran merken Sie, dass die 40 fehlt?

Stufe 3 Wie viele Zahlen müssen Sie bei 1000 sortierten Zahlen höchstens ansehen? Bei einer Million?

Auflösung: Binäre Suche von Hand

Stufe 1 In `3 8 15 23 42 57 91` die 15 suchen: **23, 8, 15**

Stufe 2 In `2 4 7 11 16 22 29 37 46 56 67 79 92 106 121` die 40 suchen: **37, 79, 56, 46**. Links von der 46 ist nichts mehr übrig: Die 40 fehlt.

Stufe 3 Höchstens wie viele Blicke bei 1000 Zahlen, bei einer Million? **10 und 20.** Jeder Blick halbiert: 2 hoch 10 = 1024, 2 hoch 20 ≈ 1 Million.

Laufzeit

Die Form der Kurve zählt

Gefragt	Wie wächst die Zahl der Schritte, wenn die Eingabe wächst?
Egal	Konstanten: n und n/2 sind beide linear.
Egal	die Basis des Logarithmus: O(log n).

Laufzeitklassen

Klasse	Beispiel
$O(n^2)$	strlen in der Bedingung
$O(n \log n)$	nächste Woche
$O(n)$	lineare Suche, n = strlen(s)
$O(\log n)$	binäre Suche
$O(1)$	kisten[3] lesen

Der beste Fall: Ω

O	höchstens: Obergrenze, für den schlechtesten Fall
Ω	mindestens: Untergrenze, für den besten Fall
Lineare Suche	O(n), Ω(1)
Binäre Suche	O(log n), Ω(1)

search.c

Sie sind das Programm


```c
int numbers[] = {20, 500, 10, 5, 100, 1, 50};

int n = get_int("Zahl: ");
for (int i = 0; i < 7; i++)
{
    if (numbers[i] == n)
    {
        printf("Gefunden\n");
    }
    else
    {
        printf("Nicht gefunden\n");
    }
}
```

search.c


```
$ ./search
Zahl: 50
Nicht gefunden
Nicht gefunden
Nicht gefunden
Nicht gefunden
Nicht gefunden
Nicht gefunden
Gefunden
```

Ohne `else`


```
for (int i = 0; i < 7; i++)
{
    if (numbers[i] == n)
    {
        printf("Gefunden\n");
    }
}
printf("Nicht gefunden\n");
```

```
$ ./search
Zahl: 50
Gefunden
Nicht gefunden
```

Was gibt das Programm jetzt aus?

```
for (int i = 0; i < 7; i++)
{
    if (numbers[i] == n)
    {
        printf("Gefunden\n");
        break;
    }
}
printf("Nicht gefunden\n");
```

Eingabe 50: A nur `Gefunden` · B `Gefunden` und `Nicht gefunden`

B: `break` verlässt nur die Schleife, danach geht es darunter weiter.

`return` beendet `main`

```c
for (int i = 0; i < 7; i++)
{
    if (numbers[i] == n)
    {
        printf("Gefunden\n");
        return 0;
    }
}
printf("Nicht gefunden\n");
return 1;
```

`break` verlässt die Schleife, `return` die Funktion, in `main` also das Programm.

`break` mit Merkvariable


```
bool gefunden = false;
for (int i = 0; i < 7; i++)
{
    if (numbers[i] == n)
    {
        gefunden = true;
        break;
    }
}

if (gefunden)
...
```

continue

Was gibt dieses Programm aus?

```c
int scheine[] = {20, 500, 10, 5, 100, 1, 50};

int summe = 0;
for (int i = 0; i < 7; i++)
{
    if (scheine[i] < 10)
    {
        continue;
    }
    summe += scheine[i];
}
printf("%i\n", summe);
```

A 686 · B 680 · C 530

B: 680, denn die 5 und die 1 überspringt `continue`.

`break` oder `continue`

Anweisung	beendet
continue	diesen Durchlauf
break	die Schleife (hier: 530)
return	die Funktion (in main: das Programm)

Wörter suchen

Welche Bedingung findet `Hund`?

```
string strings[] = {"Schiff", "Auto", "Hund", "Katze", "Hut"};

// A
if (strings[i] == s)

// B
if (strcmp(strings[i], s))

// C
if (strcmp(strings[i], s) == 0)
```

C ist richtig. A findet nie etwas: == taugt für Strings nicht.

B findet alles: wahr, sobald die Wörter verschieden sind.

Zwei gleiche Wörter

```
string s = get_string("s: ");    // Hallo
string t = get_string("t: ");    // Hallo
if (s == t)
{
    printf("Gleich\n");
}
```

Gibt nichts aus: == vergleicht bei Strings nicht die Zeichen.

Warum, sehen wir in ein paar Wochen.

Strings vergleichen: `strcmp`

```c
#include <string.h>

strcmp("Hund", "Hund")     // 0: gleich
strcmp("Hund", "Katze")    // negativ: "Hund" kommt vorher
strcmp("Katze", "Hund")    // positiv: "Katze" kommt später
```

Gleich heißt 0. Zeichen für Zeichen, nach ASCII.

Alles außer 0 ist wahr: Darum findet B alles.

Telefonbuch

Ein Telefonbuch mit zwei Arrays


```
string names[] = {"Office", "Dominik", "Bittle"};
string numbers[] = {"+49-951-863-2795",
                    "+49-951-863-2661",
                    "+49-951-863-2661"};

for (int i = 0; i < 3; i++)
{
    if (strcmp(names[i], name) == 0)
    {
        printf("Gefunden: %s\n", numbers[i]);
        return 0;
    }
}
```

Was passiert beim Sortieren?

Name und Nummer hängen nur über den Index zusammen.

structs

Ein eigener Datentyp: person

```
typedef struct
{
    string name;
    string number;
}
person;

person people[3];
people[0].name = "Office";
people[0].number = "+49-951-863-2795";
```

Das Telefonbuch mit `person`


```
string name = get_string("Name: ");
for (int i = 0; i < 3; i++)
{
    if (strcmp(people[i].name, name) == 0)
    {
        printf("Gefunden: %s\n", people[i].number);
        return 0;
    }
}
```

Was gehört in einen `song`?

```
typedef struct
{
    ...
}
song;
```

```
typedef struct
{
    string titel;
    string kuenstler;
    int jahr;
}
song;
```

Wie viele Zeilen gibt `playlist` aus?

`Bohemian Rhapsody` 1975 · `Flowers` 2023 · `Blinding Lights` 2019

```
for (int i = 0; i < 3; i++)
{
    if (playlist[i].jahr > 2020)
    {
        printf("%s\n", playlist[i].titel);
    }
}
```

A 1 · B 2 · C 3

A. Nur `Flowers` ist nach 2020 erschienen.

Zusammenfassung

Wer sortiert die Kisten?

Die binäre Suche
braucht sortierte Kisten.
Wie sortiert man sie,
und was kostet das?

Dies war Inf-Einf-B.

Übung 3 startet heute: Cäsar, Abgabe bis Montag, 16. November.

Übung 2 läuft: Scrabble und Lesbarkeit, Abgabe bis Montag, 9. November.

Shorts 3.1 (lineare Suche) und 3.2 (binäre Suche).